Attorney Docket No. MRNRP004

PATENT APPLICATION

TECHNIQUE FOR IMPLEMENTING FRACTIONAL INTERVAL TIMES FOR FINE GRANULARITY BANDWIDTH ALLOCATION

Inventors:

Kenneth W Brinkerhoff
27825 Perales
Mission Viejo, CA 92692
Citizen of U.S.A.

Wayne P Boese
2053A Tustin Ave.
Costa Mesa, CA 92627
Citizen of U.S.A.

Robert C Hutchins
24272 Solonica St.
Mission Viejo, CA 92691
Citizen of U.S.A.

Stanley Wong
2409 West Hall Ave.
Santa Ana, CA 92704
Citizen of U.S.A.

Assignee:

Mariner Networks, Inc.
1585 S. Manchester Avenue
Anaheim, CA 92802-2907

BEYER WEAVER & THOMAS, LLP
P.O. Box 778
Berkeley, CA 94704-0778
Telephone (510) 843-6200

TECHNIQUE FOR IMPLEMENTING FRACTIONAL INTERVAL TIMES FOR FINE GRANULARITY BANDWIDTH ALLOCATION

Inventors:

Kenneth W Brinkerhoff
27825 Perales
Mission Viejo, CA 92692
Citizen of U.S.A.

Wayne P Boese
2053A Tustin Ave.
Costa Mesa, CA 92627
Citizen of U.S.A.

Robert C Hutchins
24272 Solonica St.
Mission Viejo, CA 92691
Citizen of U.S.A.

Stanley Wong
2409 West Hall Ave.
Santa Ana, CA 92704
Citizen of U.S.A.

Assignee:

Mariner Networks, Inc.
1585 S. Manchester Avenue
Anaheim, CA 92802-2907

RELATED APPLICATION DATA

The present application claims priority under 35 USC 119(e) from U.S. Provisional Patent Application No. 60/215,558 (Attorney Docket No. MO15-1001-Prov) entitled "INTEGRATED ACCESS DEVICE FOR ASYNCHRONOUS TRANSFER MODE (ATM) COMMUNICATIONS"; filed June 30, 2000, and naming Brinkerhoff, et. al., as inventors (attached hereto as Appendix A); the entirety of which is incorporated herein by reference for all purposes.

The present application is also related to U.S. Patent Application Serial No. _____________ (Attorney Docket No. MRNRP005), entitled "CONNECTION SHAPING CONTROL TECHNIQUE IMPLEMENTED OVER A DATA NETWORK", naming Brinkerhoff, et. al., as inventors, and filed concurrently herewith; the entirety of which is incorporated herein by reference for all purposes.

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates generally to data networks, and more specifically to a technique for implementing fractional interval times for fine granularity bandwidth allocation.

Description of Related Arts

Many communication protocols such as, for example, frame relay and ATM, require that a continuous stream of bits be continuously transmitted between endpoints of a communication link. For such protocols, a variety of mechanisms exist for enabling the end point receiving the continuous bit stream to differentiate between data parcels (e.g. frames, cells, etc.) which contain meaningful data, and data parcels which do not contain meaningful data, but rather are transmitted by the transmitting end merely to satisfy the continuous bit stream requirement.

For example, in frame relay protocols, as described, for example, in the Frame Relay Forum (FRF) Reference Document FRF.1.2, July, 2000, specific patterns of flag bytes are used to indicate that a particular portion of continuous bits (forming a frame) corresponds to a "filler" frame which does not contain meaningful data. In an ATM network, such as that defined, for example, in the reference document entitled, "A Cell-based Transmission Convergence Sublayer for Clear Channel Interfaces", af-phy-0043.000, Nov. 1995, cells which contain meaningful data are referred to as data cells, and cells which do not contain meaningful data are referred to as idle cells. According to conventional ATM protocol, an ATM cell may be identified as being either a data cell or an idle by referencing the information contained in the header portion of the ATM cell.

In conventional ATM networks, each ATM transceiver performs a decision making process at continuous, fixed intervals as to whether the next cell to be transmitted is to be a data cell or an idle cell. The frequency of the fixed intervals may be determined by line modulation and framing formatting, and is typically implemented using an internal clock source. Typically, if the ATM transceiver has meaningful data to transmit, it will transmit the meaningful data using one or more data cells. However, if there is no meaningful data to be transmitted, the ATM transceiver will continuously transmit idle cells until new meaningful data is ready to be transmitted. This process is described in greater detail with respect to FIGURE 1 of the drawings.

FIGURE 1 shows a block diagram of a portion 100 of a conventional ATM network. Network portion 120 corresponds to a node of the ATM network, such as, for example, an end point of an ATM link.

As shown in FIGURE 1, network portion 120 includes ATM transceiver componentry 110 which receives data from a plurality of different clients or flows. Each client may be associated with a specific line rate which may be different than the line rate used by the WAN service provider 150. For example, as shown in FIGURE 1, lines 109, 101C and 101D may each correspond to an ATM E1 communication line in which data is transmitted at 2.048 Mbps. Lines 101A and 101B may each correspond to a T1 communication line in which data is transmitted at 1.544 Mbps.

Each of the lines 101A-D is associated with a respective process or client flow. As shown in FIGURE 1, each client flow has an associated buffer for storing output data to be transmitted by output transceiver 114 over line 109. For example, Process A1 (not shown) uses buffer 111 to store outgoing data generated by Process A1, which is eventually to be transmitted by output transceiver 114 over line 109.

As shown in the example of FIGURE 1, output data from the client different processes are queued into buffers (e.g. 102A, 102B) to await scheduling. Since the rate of data queued into each buffer may vary, depending upon the bit rate associated with each process, a plurality of different schedulers (e.g. 104A, 104B) are typically used to schedule the output data. Typically each scheduler is responsible for scheduling output data associated with a specific bit rate. The schedulers prioritize the output data from the different client processes, and enqueue the scheduled output data cells into the output transceiver buffer 112 to await transmission over communication line 109.

According to conventional techniques, the scheduling algorithm performed by a scheduler is based upon quality of service (QoS) parameters and a local time base, which is typically generated by a local clock source. Since each of the client processes may be associated with different bit rates of data transmission, a plurality of different schedulers are typically employed to handle the scheduling of data cells corresponding to the different bit rates. For example, shown in FIGURE 1, it is assumed that data lines 101A and 101B have the same line rate, and therefore are scheduled by Line Rate A Scheduler 104A. Data lines 101C and 101D also have the line rate, which is different from that of data lines 101A-B, and therefore are handled by Line Rate B Scheduler 104B.

As shown in FIGURE 1, each scheduler is driven by a separate clock source (e.g. 106A, and 106B) which has been designed specifically for the particular line rate or bit rate associated with the processes which that scheduler services. For example, Scheduler A 104A is driven by a first local time base generated by clock 106A, which has been specifically designed to work with the line rate associated with lines 101A, 101B. Scheduler B 104B is driven by a second local time base generated by clock 106B, which has been specifically designed to work with the line rate associated with lines 101C, 101D. Each scheduler will clock in data from its respective input buffers at a different rate, and enqueue the clocked data cells into the output transceiver FIFO 112.

It is noted that the provisioning of a separate clock source for each scheduler significantly increases the cost and complexity of the scheduling system. Moreover, using the above-described scheduling technique, each different line rate necessitates the provisioning of additional scheduling logic for servicing flows associated with that particular line rate. Additionally, since the clock sources driving each of the schedulers are typically not synchronized to each other, a non-uniform pattern of data/idle cells may be transmitted from the output transceiver 114 over line 109, thereby hindering system analysis and testing operations.

Accordingly, it will be appreciated that there exists a continual need to improve upon data parcel scheduling techniques in order to overcome conventional limitations and undesirable aspects, and to accommodate new and emerging network applications and technologies.

SUMMARY OF THE INVENTION

According to various embodiments of the present invention, a method and computer program product are disclosed for scheduling data parcels from at least one client process to be output for transmission over a first communication line having an associated first bit rate. The at least one client process includes a first client process having an associated bit rate. According to specific embodiments, the at least one client process may also include additional client processes, each having a respective, associated bit rate. A plurality of data parcels associated with the client processes are identified by a scheduler. The scheduler performs scheduling operations and selects specific client data parcels to be included in an output stream provided to physical layer logic for transmission over the first communication line. An appropriate ratio of "filler" data parcels to be inserted into the output stream is determined. In one embodiment, the ratio of "filler" data parcels to be inserted into the output data stream is sufficient to cause a bit rate of the output stream to be substantially equal to the bit rate associated with the first communication line. According to specific embodiments, the "filler" data parcels correspond to disposable data parcels which do not include meaningful data. For example, in one implementation, the "filler" data parcels correspond to idle ATM cells.

An alternate embodiment of the present invention is directed to a system for scheduling data parcels from at least one client process to be output for transmission over a first communication line having an associated first bit rate. The at least one client process includes a first client process having an associated bit rate. According to specific embodiments, the at least one client process may also include additional client processes, each having a respective, associated bit rate. The system comprises a scheduler adapted to identify incoming client data parcels from one or more of the client processes, and to generate an output stream of data parcels to be provided to the physical layer logic for transmission over the first communication line. The scheduler is configured to generate "filler" data parcels which include non-meaningful data. The scheduler is also configured to determine an appropriate ratio of filler data parcels to be inserted into the scheduler output stream. In one embodiment, the ratio of "filler" data parcels to be inserted into the output data stream is sufficient to cause a bit rate of the output stream to be substantially equal to the bit rate associated with the first communication line.

According to specific embodiments, the output stream is generated by the scheduler, and may include client data parcels (e.g. data parcels originating from the client processes) as well as "filler" data parcels. Additionally, according to one embodiment, the output stream includes a uniform pattern of client data parcels and filler data parcels which may repeat on a periodic basis.

Additionally, according to specific embodiments, the scheduler is devoid of an internal clock source, and may perform scheduling operations based upon ratios of client and "filler" data parcels, rather than on an internal time base or reference signal. Further, according to a specific embodiment, the scheduler includes an ATM cell switch and/or quality of service (QOS) scheduling logic.

Additional objects, features and advantages of the various aspects of the present invention will become apparent from the following description of its preferred embodiments, which description should be taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 shows a block diagram of a portion 100 of a conventional ATM network.

FIGURE 2 shows a block diagram of a specific embodiment of a portion of a data network which may be used for implementing the technique of the present invention.

FIGURE 3 shows a specific embodiment of a scheduler 330 which may be used for implementing the scheduling technique of the present invention.

FIGURE 4 shows a flow diagram of a Ratio Computation Procedure 400 in accordance with a specific embodiment of the present invention.

FIGURE 5 shows an example of a RCC Table 500 in accordance with a specific embodiment of the present invention.

FIGURE 6A shows a specific implementation of a Client Cell Interval Table 650 which may be used for implementing the scheduling technique of the present invention.

FIGURE 6B illustrates an output stream transmitted by the scheduler 204 in accordance with a specific embodiment of the present invention.

FIGURE 7 shows a specific embodiment of a network device 60 suitable for implementing various techniques of the present invention.

FIGURE 8 shows a block diagram of various inputs/outputs, components and connections of a system 800 which may be used for implementing various aspects of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIGURE 2 shows a block diagram of a specific embodiment of a portion of a data network which may be used for implementing the scheduling technique of the present invention. As explained in greater detail below, the scheduling technique of the present invention provides a new mechanism for accurately scheduling different data flows associated with different bit rates based upon desired data/idle cell patterns. In order to gain a better understanding of the differences between the scheduling technique of the present invention and conventional scheduling techniques, it is helpful to review the process by which idle cells are generated in conventional scheduling techniques.

According to conventional scheduling techniques as described, for example, in FIGURE 1, scheduler 104A will clock data cells at fixed intervals (which are driven by clock 106A) from buffers 102A to the output transceiver componentry 110. Similarly, scheduler 104B clocks the data cells at fixed intervals (which are driven by clock 106B) from buffers 102B to the output transceiver componentry 110. If, at any given time, there are no data cells queued in either buffers 102A or 102B, then the respective scheduler servicing the empty buffers will be idle.

It will be appreciated that conventional schedulers are not configured to generate idle cells. Rather, according to conventional scheduling techniques, the generation of idle cells is handled by the physical layer such as the output transceiver componentry 110. For example, if there are data cells queued in the transmitter FIFO 112, the output transceiver 114 will dequeue the cells from buffer 112 at fixed periodic intervals, and transmit the dequeued data cells over line 109. However, if the output transceiver determines that it is time to transmit a next ATM cell over line 109, and the buffer 112 is empty, the output transceiver will generate and transmit an idel cell over line 109 at the designated time. Thus, it will be appreciated that, in conventional scheduling techniques, the ATM transceiver is responsible for the generation of idle cells. Additionally, it will be appreciated that, since the clock sources driving each of the schedulers are typically not synchronized, a non-uniform pattern of data/idle cells is transmitted from the output transceiver 114 over line 109. Such a non-uniform pattern of data/idle cells makes it difficult to perform system analysis measurements for verifying proper operation of the various system components.

In contrast to conventional scheduling techniques which utilize an internal time base for scheduling and clocking output data from different client processes into the output transceiver buffer, the scheduling technique of the present invention determines an appropriate ratio of data cells and idle cells for each client process, and effectively achieves proper scheduling and timing functionality by periodically inserting an appropriate number of idle cells into the output data stream associated with a selected client process.

According to a specific embodiment as shown, for example, in FIGURE 2 of the drawings, the scheduler 204 is configured to service a plurality of different client processes which may have different associated line rates. The client processes store their output data cells in output buffers 202A, 202B. As shown in the embodiment of FIGURE 2, the scheduler 204 includes a ratio computation component (RCC) 206 which may be configured to perform functions for determining the appropriate ratio of idle cells to be inserted into the output data stream(s) of selected client processes in order to achieve a desired timing relationship of data/idle cells which may then be passed to the output transceiver circuitry 220 for transmission over line 209.

Using the functionality of the ratio computation component 206, the scheduler 204 may begin generating an output data stream on line 205. According to specific implementation, the scheduler 204 may be configured to have an output rate which is sufficiently fast enought to ensure that the output transceiver buffer is never empty. In this way, the physical layer (e.g. transceiver componentry 220) is prevented from generating and inserting idle cells into the output data stream. In one implementation, the output data stream on line 205 preferably has an effective line rate equal to that of line 209. Additionally, according to specific implementations of the present invention, the output data stream on line 205 will include not only data cells from each of the client processes 201A-D, but will also include an appropriate number or ratio of idle cells which have been inserted into the output data stream 205 to thereby cause line 205 to have an effective line rate equal to that of line 209. These features are described in greater detail below with respect to FIGURES 4 and 5 of the drawings.

FIGURE 4 shows a flow diagram of a Ratio Computation Procedure 400 in accordance with a specific embodiment of the present invention. In one implementation, the Ratio Computation Procedure 400 may be implemented by the scheduler 204 of FIGURE 2. For purposes of illustration, the flow diagram of FIGURE 4 will now be described in greater detail using the example of FIGURES 6A and 6B.

In the example of FIGURE 6A, it is assumed that two different client processes, namely Client 1 (C1) and Client 2 (C2), are generating output data which is to be transmitted by the output transceiver circuitry 220 (FIGURE 2) over line 209. Additionally, in this example, it is assumed that Client 1 is connected to line 201A (FIGURE 2) which has a line rate of Line Rate A, and Client 2 is connected to line 201C, which has a line rate of Line Rate B. Further, it is assumed that the line rate corresponding to line 209 is represented as Line Rate C.

FIGURE 6A shows a specific implementation of a Client Cell Interval Table 650 which may be used for implementing the scheduling technique of the present invention. As shown in table 650, each client process or flow may have an associated cell interval (I_i) value which represents how often a data cell from a particular flow is to be transmitted over line 209. For example, as shown in FIGURE 6A, client flow C1 has an associated interval value of I1 = 2.5, meaning that a new data cell from client flow C1 is to be scheduled once every 2.5 ATM cells which are transmitted over line 209. Similarly, client flow C2 has an associated interval value of I2 = 2.75, meaning that a new data cell from client flow C2 is to be scheduled once every 2.75 ATM cells which are transmitted over line 209. According to a specific implementation, the cell interval value may be defined as an integer, a fixed point integer, a floating point number, etc.

According to a specific implementation, each flow associated with a specific client may have an associated cell interval value [i]. According to different embodiments, computation of the cell interval value for each flow may be determined based upon several factors such as, for example, QoS, line rate of the client flow (herein referred to as the "input line rate"), line rate of the service provider (herein referred to as the "output line rate"), etc. For example, if line 201A (which services client flow C1) has a line rate of 1.5 Mbps, and the line rate of the service provider line 209 is 3.0 Mbps, then the cell interval value for client flow C1 may be calculated according to: 3Mbps/1.5Mbps = 2, which means that client flow C1 has the potential to transmit a data cell every two ATM cells which are transmitted over line 209. Similarly, if the line rate for line 201C (servicing client flow C2) is equal to 1.0 Mbps, then the cell interval value for client C2 would be equal to 3Mbps/1Mbps = 3, meaning that client flow C2 has the potential to transmit a data cell for every third ATM cell which is transmitted over line 209. It will be appreciated that the cell interval value for any selected flow may also be adjusted based upon the QoS value associated with that flow.

According to different embodiments of the present invention, the cell interval value for each flow may either be statically or dynamically determined. For example, according to a specific implementation, as shown, for example, in FIGURE 7, calculation of the different cell interval values for each flow may be calculated by a processor such as processor 62A or 62B. When a particular line card is electrically coupled to the system 60 of FIGURE 7, the respective line rates of the ports residing on that particular line card may be stored in line card memory 72. This data may then be accessed by a processor such as 62A or 62B, which uses the port line rate information to calculate a respective cell interval value for each port. The cell interval values may then be stored locally in memory such as, for example, in CPU memory 61 or in system memory 65. Since data from each client flow is associated with a respective port, the cell interval value associated with a particular client flow may be equal to the cell interval rate for the associated port, adjusted by the QoS parameter(s) associated with that client flow. Once the cell interval value for a specific client flow has been determined, that value may be stored in table 650 (FIGURE 6A), which may reside, for example, in processor memory or system memory (FIGURE 7).

In the example of FIGURE 6A, it is assumed that the client flow C1 has a cell interval value I1 = 2.5, and client flow C2 has a cell interval value I2 = 2.75. Using the example of FIGURE 6A, the Ratio Computation Procedure 400 of FIGURE 4 will now be described in order to derive the output stream 602 illustrated in FIGURE 6B, which, according to a specific implementation, illustrates an output stream transmitted by the scheduler 204 on line 205 of FIGURE 2. According to a specific implementation, this output stream is identical to the output stream transmitted by output transceiver 214 over line 209.

Initially, as shown at 402 of FIGURE 4, a number of parameters corresponding the each of the selected client flows are initialized. In the example of FIGURE 6A, it is assumed that the Ratio Computation Procedure 400 will be used to schedule data slots for 2 client processes, namely C1 and C2. However, it will be appreciated that any desired number of client processes or flows maybe scheduled using at least one scheduler which has been implemented in accordance with the technique of the present invention.

As shown at 402, the cell interval value (I_I) for each client flow is determined or retrieved. Additionally, the next calculated data cell interval value (N_I) for each client flow is set equal to zero. Thus, for example, a first variable N1 (corresponding to client flow C1) may be initialized and set equal to zero, and a second variable N2 (corresponding to client flow C2) may also be initialized and set equal to zero. According to a specific implementation, the variable N_I may be defined as a fixed point fraction. The parameter N_I is described in greater detail below. Additionally, at 402, the total number (T) of cell intervals which have elapsed since the start of the Ratio Computation Procedure is set equal to zero. According to a specific implementation, the parameter T may be represented as an integer which keeps track of the total number of ATM cells which have been transmitted over line 209 since the start of the Ratio Computation Procedure 400.

According to a specific embodiment of the present invention, at least some of the initialized variables of the Ratio Computation Procedure 400 may be stored in a table such as, for example, RCC Table 500 of FIGURE 5. As shown in FIGURE 5, the RCC Table 500 may include a plurality of entries (e.g. 501, 503, 505), wherein each entry includes a first field 502 for identifying a specific client flow, a second field 504 for identifying a particular cell interval value (I_I) associated with that flow, and a third field 506 for identifying the next calculated data cell interval value (N_I) for that flow. In the present example, the RCC Table 500 may include the following values at the cell interval T = 0:

Client ID	I Value	N Value
C1	2.5	0
C2	2.75	0

After initialization has been performed, a determination is made (404) as to whether any information is allowed to be sent from the scheduler 204 to the output transceiver componentry 220. According to a specific implementation, this determination made be made by checking to see whether the output transceiver buffer 212 is full. Assuming that the output transceiver buffer 212 is not full, a determination is then made (412) as to whether every integer value of N_I for each client flow is greater than the current value of T.

Since N1 and N2 are each less than or equal to T at this point, flow continues procedural block 414 wherein the client flow having the smallest I_I value is selected (414). In the present example, this operation would result in the selecting of client C1 since the value I1 = 2.5 (corresponding to Client C1) is less than the value I2 = 2.75 (corresponding to Client C2). A next data cell for the selected client process (e.g. C1) is then transmitted by the scheduler to the output transceiver circuitry 220. Thus, as shown in FIGURE 6B, the cell which is transmitted by scheduler 204 at time T = 0 corresponds to a data cell associated with client flow C1. According to a specific implementation, the transmitted data cell may be obtained from the appropriate client flow buffer (e.g. 221) corresponding to the selected client flow.

According to a specific embodiment, if there is no data to be dequeued from the selected client flow buffer, a different client flow may be selected from the set of client flows satisfying the criteria integer [N_I] <= T, where the newly selected client has the next smallest I_I value.

Returning to FIGURE 4, after the data cell for the selected client flow has been sent to the output transceiver circuitry 220, the value N1 is incremented (418) by the value I1. In the present example, the new value for N1 will be 0 + 2.5 = 2.5. This updated value for N1 may be stored in an appropriate location at the RCC Table 500 (FIGURE 5). Thereafter, the value T is incremented (420). According to the embodiment of FIGURE 4, the value T is incremented by one, resulting in a new value of T = 1. Thereafter, flow of the Ratio Computation Procedure 400 continues at procedural block 404.

During each iteration of the Ratio Computation Procedure 400, a new cell is sent from the scheduler 204 to the output transceiver circuitry 220. According to a specific implementation, the different type of cells which may be transmitted by the scheduler 204 to the output transceiver circuitry 220 may include data cells from any of the plurality of client flows which that scheduler services, or idle cells. By inserting an appropriate ratio of idle cells into the data stream output by the scheduler on line 205, a plurality of different client flows associated with different line rates may be serviced by a single scheduler. Moreover, unlike conventional scheduling techniques, the scheduler of the present invention need not include additional clock circuitry and/or logic for clocking data flows corresponding to different line rates.

Returning to FIGURE 4, at the beginning of the next iteration of the Ratio Computation Procedure 400, the values of the various parameters are as follows: I1 = 2.5, I2 = 2.75, N1 = 2.5, N2 = 0, and T = 1.

Assuming that data is allowed to be sent to the output transceiver componentry 220, the integer values of N1 and N2 are compared to the value T in order to determine whether each of these values exceeds the value of T. In the present example, the value of N2 is 0, which is less than the value of T (since T = 1). Therefore, the Ratio Computation Procedure continues at 414, wherein the client flow with the smallest I_I value is selected from a set of client flows whose integer values of N_I are less than or equal to T. In the present example, client flow C2 will be selected at operation 414.

Thereafter, a data cell for client C2 may be dequeued from its corresponding output buffer (e.g. 215) and is transmitted to the output transceiver circuitry 220 via line 205. Thus, as shown in FIGURE 6B, a data cell corresponding to the client flow C2 is transmitted on line 205 at the relative "time" value defined by T = 1. Thereafter, at 418, the value N2 is incremented to N2 = 2.75, and the value T is incremented to T = 2.

At the beginning of the next iteration of the Ratio Computation Procedure, we find N1 = 2.5, N2 = 2.75, and T = 2. Since the integer values of N1 and N2 are each not greater than T, the Ratio Computation Procedure will select (414) client flow C1 at 414, and transmit a data cell from client C1 to the output transceiver circuitry 220 via line 205. Thus, as shown in FIGURE 6B, a data cell from the client C1 flow will be sent to the output transceiver circuitry at time T = 2. Thereafter, the value N1 will be incremented to N1 = 5 and the value T will be incremented to T = 3.

At the beginning of the next iteration of the Ratio Computation Procedure 400, we find N1 = 5, N2 = 2.75, and T = 3. Based on these parameter values, the next cell sent by the scheduler 204 to the output transceiver circuitry 220 will be a data cell corresponding to the C2 client flow. As illustrated in FIGURE 6B, a data cell from the client C2 flow will be sent to the output transceiver circuitry at time **T** = 3. Thereafter, the value N2 will be incremented to N1 = 5.5 and the value T will be incremented to T=4.

At the beginning of the next iteration of the Ratio Computation Procedure 400 we find N1 = 5.0, N2 = 5.5, and T = 4. At operation 412, the Ratio Computation Procedure will determine that each of the integer values of N1 and N2 is greater than T (412). As a result, the ration computation procedure will send (417) an idle cell via line 205 to the output transceiver circuitry 220, as shown, for example, in FIGURE 6B at T=4. Thereafter, the value of T is incremented to T = 5.

According to a specific embodiment, each successive of iteration of the Ratio Computation Procedure results in a new cell (e.g. either a data cell or an idle cell) being sent by the scheduler 204 to the output transceiver circuitry. As shown by the pattern of the output stream 602 of FIGURE 6B, one idle cell will be inserted into the scheduler output data stream following the scheduling of two C1 data cells and two C2 data cells, thereby resulting in an initial pattern of C1-C2-C1-C2-I. As time progresses the pattern may change with idle cells interspersed to reflect the different ratios of C1 and C2. For example, at a later iteration the pattern generated in this example will be C1-C2-C1-I-C2.

Additionally, as shown in the example of FIGURE 6B, it can be seen that the pattern of data/idle cells transmitted over line 209 will eventually repeat itself. Thus, it will be appreciated that the scheduling technique of the present invention provide an advantage over conventional scheduling techniques in that a pattern of data/idle cells transmitted over line 209 may be uniform, periodic and/or predictable. In contrast, conventional scheduling techniques such as that described previously with respect to FIGURE 1 generate an output data stream in which the pattern of idle/data cells is not predictable (due primarily to the fact that the different clock sources in each of the conventional schedulers are typically not synchronized).

It will also be appreciated that, using the scheduling technique of the present invention, no additional idle cells need be added by the physical layer or output transceiver circuitry 220. As a result, according to at least one embodiment, the pattern of data cells and idle cells transmitted by the output transceiver 214 over line 209 may exactly match the data/idle cell pattern on line 205, as shown, for example, in FIGURE 6B.

Further, it will be appreciated that the scheduling technique of the present invention provides a number of additional advantages which are not realized by conventional scheduling techniques. For example, according to one implementation, the scheduling technique of the present invention provides for a uniform output data flow from the ATM transceiver, wherein the pattern of data/idle cells conforms with a cyclical or periodic pattern. Additionally, according to a specific embodiment, the scheduler of the present invention may perform its scheduling functions without requiring the use of an independent or separate clock source such as those required in conventional schedulers (described previously with respect to FIGURE 1). The elimination of the clock source circuitry and accompanying logic results in a simplified scheduler design and a significant reduction in manufacturing costs.

Another difference between the scheduling technique of the present invention and that of conventional scheduling techniques is that unlike conventional scheduling techniques which base their operation on an internal time base, the scheduling technique of the present invention bases its operation on idle/data cell ratio or patterns, rather than on an internal time base.

Yet another difference between the scheduling technique of the present invention and those of conventional scheduling techniques is that the scheduler of the present invention may be configured or designed to generate idle cells. In contrast, conventional schedulers typically do not provide such functionality since the physical layer or output transceiver circuitry already includes such functionality.

FIGURE 3 shows a specific embodiment of a scheduler 330 which may be used for implementing the scheduling technique of the present invention. According to specific embodiments, the scheduling technique of the present invention may be implemented via hardware such as that shown, for example, in FIGURE 3, and/or software such as that described, for example, in FIGURE 4. As shown in the embodiment of FIGURE 3, scheduling circuitry 330 includes a QoS scheduler 332 which includes the appropriate logic for implementing the ratio computation component functionality of the present invention. The QoS scheduler 332 may be responsible for dequeueing data cells from the plurality of client flow buffers 370, prioritizing and scheduling output of each of the data cells in accordance with line rate and QoS parameters, passing the output data cells via line 335 to a logical OR functionality block 334. According to a specific implementation, the logical OR functionality block 334 may be configured to perform a logical OR function using data received from input lines 335 and 337. Input line 337 is connected to an idle cell generator 336 which continuously generates idle cells. If the QoS scheduler 332 outputs a data cell on line 335, the logical or functionality block 334 will pass the data cell to the output transceiver buffer 312 via line 305. However, during time when the QoS scheduler is not transmitting data cells on line 335, logical OR block 334 will continuously transmit idle cells over line 305 to the output transceiver buffer 312. According to a specific implementation, if the scheduler 332 determines that an idle cells should be transmitted to the output transceiver buffer, it may be configured to not transmit any cells over line 335 during a specific time period.

It will be appreciated that overflows may occur while implementing specific aspects of the present invention. Accordingly, comparison logic should preferably take this into account to maintain proper order in the output transceiver queue and to make meaningful comparisons against relative port time (T).

System Configurations

Referring now to FIGURE 7, a network device 60 suitable for implementing the scheduling techniques of the present invention includes a master central processing unit (CPU) 62A, interfaces 68, and various buses 67A, 67B, 67C, etc., among other components. According to a specific implementation, the CPU 62A may correspond to the eXpedite ASIC, manufactured by Mariner Networks, of Anaheim, California.

Network device 60 is capable of handling multiple interfaces, media and protocols. In a specific embodiment, network device 60 uses a combination of software and hardware components (e.g., FPGA logic, ASICs, etc.) to achieve high-bandwidth performance and throughput (e.g., greater than 6 Mbps), while additionally providing a high number of features generally unattainable with devices that are predominantly either software or hardware driven. In other embodiments, network device 60 can be implemented primarily in hardware, or be primarily software driven.

When acting under the control of appropriate software or firmware, CPU 62A may be responsible for implementing specific functions associated with the functions of a desired network device, for example a fiber optic switch or an edge router. In another example, when configured as a multi-interface, protocol and media network device, CPU 62A may be responsible for analyzing, encapsulating, or forwarding packets to appropriate network devices. Network device 60 can also include additional processors or CPUs, illustrated, for example, in FIGURE 7 by CPU 62B and CPU 62C. In one implementation, CPU 62B can be a general purpose processor for handling network management, configuration of line cards, FPGA logic configurations, user interface configurations, etc. According to a specific implementation, the CPU 62B may correspond to a HELIUM Processor, manufactured by Virata Corp. of Santa Clara, California. In a different embodiment, such tasks may be handled by CPU62A, which preferably accomplishes all these functions under partial control of software (e.g., applications software and operating systems) and partial control of hardware.

CPU 62A may include one or more processors 63 such as the MIPS, Power PC or ARM processors. In an alternative embodiment, processor 63 is specially designed hardware (e.g., FPGA logic, ASIC) for controlling the operations of network device 60. In a specific embodiment, a memory 61 (such as non-persistent RAM and/or ROM) also forms part of CPU 62A. However, there are many different ways in which memory could be coupled to the system. Memory block 61 may be used for a variety of purposes such as, for example, caching and/or storing data, programming instructions, etc.

According to a specific embodiment, interfaces 68 may be implemented as interface cards, also referred to as line cards. Generally, the interfaces control the sending and receiving of data packets over the network and sometimes support other peripherals used with network device 60. Examples of the interfaces that may be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, IP interfaces, etc. In addition, various ultra high-speed interfaces can be provided such as fast Ethernet interfaces, Gigabit Ethernet interfaces, ATM interfaces, HSSI interfaces, POS interfaces, FDDI interfaces and the like. Generally, these interfaces include ports appropriate for communication with appropriate media. In some cases, they also include an independent processor and, in some instances, volatile RAM. The independent processors may control communications intensive tasks such as data parcel switching, media control and management, framing, interworking, protocol conversion, data parsing, etc. By providing separate processors for communications-intensive tasks, these interfaces allow the main CPU 62A to efficiently perform routing computations, network diagnostics, security functions, etc. Alternatively, CPU 62A may be configured to perform at least a portion of the above-described functions such as, for example, data forwarding, communication protocol and format conversion, interworking, framing, data parsing, etc.

In a specific embodiment, network device 60 is configured to accommodate a plurality of line cards 70. At least a portion of the line cards are implemented as hot-swappable modules or ports. Other line cards may provide ports for communicating with the general-purpose processor, and may be configured to support standardized communication protocols such as, for example, Ethernet or DSL. Additionally, according to one implementation, at least a portion of the line cards may be configured to support Utopia and/or TDM connections.

Although the system shown in FIGURE 7 illustrates one specific network device of the present invention, it is by no means the only network device architecture on which the present invention can be implemented. For example, an architecture having a single processor that handles communications as well as routing computations, etc., may be used. Further, other types of interfaces and media could also be used with the network device such as TI, E1, Ethernet or Frame Relay.

According to a specific embodiment, network device 60 may be configured to support a variety of different types of connections between the various components. For illustrative purposes, it will be assumed that CPU 62A is used as a primary reference component in device 60. However, it will be understood that the various connection types and configurations described below may be applied to any connection between any of the components described herein.

According to a specific implementation, CPU 62A supports connections to a plurality of Utopia lines. As commonly known to one having ordinary skill in the art, a Utopia connection is typically implemented as an 8-bit connection which supports standardized ATM protocol. In a specific embodiment, the CPU 62A may be connected to one or more line cards 70 via Utopia bus 67A and ports 69. In an alternate embodiment, the CPU 62A may be connected to one or more line cards 70 via point-to-point connections 51 and ports 69. The CPU 62A may also be connected to additional processors (e.g. 62B, 62C) via a bus or point-to-point connections (not shown). As described in greater detail below, the point-to-point connections may be configured to support a variety of communication protocols including, for example, Utopia, TDM, etc.

As shown in the embodiment of FIGURE 7, CPU 62A may also be configured to support at least one bi-directional Time-Division Multiplexing (TDM) protocol connection to one or more line cards 70. Such a connection may be implemented using a TDM bus 67B, or may be implemented using a point-to-point link 51.

In a specific embodiment, CPU 62A may be configured to communicate with a daughter card (not shown) which can be used for functions such as voice processing, encryption, or other functions performed by line cards 70. According to a specific implementation, the communication link between the CPU 62A and the daughter card may be implemented using a bi-directional TDM connection and/or a Utopia connection.

According to a specific implementation, CPU 62B may also be configured to communicate with one or more line cards 70 via at least one type connection. For example, one connection may include a CPU interface that allows configuration data to be sent from CPU 62B to configuration registers on selected line cards 70. Another connection may include, for example, an EEPROM arrow interface to an EEPROM memory 72 residing on selected line cards 70.

Additionally, according to a specific embodiment, one or more CPUs may be connected to memories or memory modules 65. The memories or memory modules may be configured to store program instructions and application programming data for the network operations and other functions of the present invention described herein. The program instructions may specify an operating system and one or more applications, for example. Such memory or memories may also be configured to store configuration data for configuring system components, data structures, or other specific non-program information described herein.

Because such information and program instructions may be employed to implement the systems/methods described herein, the present invention relates to machine-readable media that include program instructions, state information, etc. for performing various operations described herein. Examples of machine-readable media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media such as floptical disks; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory devices (ROM), Flash memory PROMS, random access memory (RAM), etc.

In a specific embodiment, CPU 62B may also be adapted to configure various system components including line cards 70 and/or memory or registers associated with CPU 62A. CPU 62B may also be configured to create and extinguish connections between network device 60 and external components. For example, the CPU 62B may be configured to function as a user interface via a console or a data port (e.g. Telnet). It can also perform connection and network management for various protocols such as Simple Network Management Protocol (SNMP).

FIGURE 8 shows a block diagram of various inputs/outputs, components and connections of a system 800 which may be used for implementing various aspects of the present invention. According to a specific embodiment, system 800 may correspond to CPU 62A of FIGURE 7.

As shown in the embodiment of FIGURE 8, system 800 includes cell switching logic 810 which operates in conjunction with a scheduler 806. In one implementation, cell switching logic 810 is configured as an ATM cell switch. In other implementations, switching logic block 810 may be configured as a packet switch, a frame relay switch, etc.

Scheduler 806 provides quality of service (QoS) shaping for switching logic 810. For example, scheduler 806 may be configured to shape the output from system 800 by controlling the rate at which data leaves an output port (measured on a per flow/connection basis). Additionally, scheduler 806 may also be configured to perform policing functions on input data. Additional details relating to switching logic 810 and scheduler 806 are described below.

As shown in the embodiment of FIGURE 8, system 800 includes logical components for performing desired format and protocol conversion of data from one type of communication protocol to another type of communication protocol. For example, the system 800 may be configured to perform conversion of frame relay frames to ATM cells and vice-versa. Such conversions are typically referred to as interworking. In one implementation, the interworking operations may be performed by Frame/Cell Conversion Logic 802 in system 800 using standardized conversion techniques as described, for example, in the following reference documents, each of which is incorporated herein by reference in its entirety for all purposes

ATM Forum

(1) "B-ICI Integrated Specification 2.0", af-bici-0013.003, Dec. 1995

(2) "User Network Interface (UNI) Specification 3.1", af-uni-0010.002, Sept. 1994

(3) "Utopia Level 2, v1.0", af-phy-0039.000, June 1995

(4) "A Cell-based Transmission Convergence Sublayer for Clear Channel Interfaces", af-phy-0043.000, Nov. 1995

Frame Relay Forum

(5) "User-To-Network Implementation Agreement (UNI)", FRF.1.2, July 2000

(6) "Frame Relay/ATM PVC Service Interworking Implementation Agreement", FRF.5, April 1995

(7) "Frame Relay/ATM PVC Service Interworking Implementation Agreement", FRF.8.1, Dec. 1994

ITU-T

(8) "B-ISDN User Network Interface - Physical Layer Interface Specification", Recommendation I.432, March 1993

(9) "B-ISDN ATM Layer Specification", Recommendation I.361, March 1993

As shown in the embodiment of FIGURE 8, system 800 may be configured to include multiple serial input ports 812 and multiple parallel input ports 814. In a specific embodiment, a serial port may be configured as an 8-bit TDM port for receiving data corresponding to a variety of different formats such as, for example, Frame Relay, raw TDM (e.g., HDLC, digitized voice), ATM, etc. In a specific embodiment, a parallel port, also referred to as a Utopia port, is configured to receive ATM data. In other embodiments, parallel ports 814 may be configured to receive data in other formats and/or protocols. For example, in a specific embodiment, ports 814 may be configured as Utopia ports which are able to receive data over comparatively high-speed interfaces, such as, for example, E3 (35 megabits/sec.) and DS3 (45 megabits/sec.).

According to a specific embodiment, incoming data arriving via one or more of the serial ports is initially processed by protocol conversion and parsing logic 804. As data is received at logic block 804, the data is demultiplexed, for example, by a TDM multiplexer (not shown). The TDM multiplexer examines the frame pulse, clock, and data, and then parses the incoming data bits into bytes and/or channels within a frame or cell. More specifically, the bits are counted to partition octets to determine where bytes and frames/cells start and end. This may be done for one or multiple incoming TDM datapaths. In a specific embodiment, the incoming data is converted and stored as sequence of bits which also include channel number and port number identifiers. In a specific embodiment, the storage device may correspond to memory 808, which may be configured, for example, as a one-stack FIFO.

According to different embodiments, data from the memory 808 is then classified, for example, as either ATM or Frame Relay data. In other preferred embodiments, other types of data formats and interfaces may be supported. Data from memory 808 may then be directed to other components, based on instructions from processor 816 and/or on the intelligence of the receiving components. In one implementation, logic in processor 816 may identify the protocol associated with a particular data parcel, and assist in directing the memory 808 in handing off the data parcel to frame/cell conversion logic 802.

In the embodiment of FIGURE 8, frame relay/ATM interworking may be performed by interworking logic 802 which examines the content of a data frame. As commonly known to one having ordinary skill in the art of network protocol, interworking involves converting address header and other information in from one type of format to another. In a specific embodiment, interworking logic 802 may perform conversion of frames (e.g. frame relay, TDM) to ATM cells and vice versa. More specifically, logic 802 may convert HDLC frames to ATM Adaptation Layer 5 (AAL 5) protocol data units (PDUs) and vice versa. Interworking logic 802 also performs bit manipulations on the frames/cells as needed. In some instances, serial input data received at logic 802 may not have a format (e.g. streaming video), or may have a particular format (e.g., frame relay header and frame relay data).

In at least one embodiment, the frame/cell conversion logic 802 may include additional logic for performing channel grooming. In one implementation, such additional logic may include an HDLC framer configured to perform frame delineation and bit stuffing. As commonly known to one having ordinary skill in the art, channel grooming involves organizing data from different channels in to specific, logical contiguous flows. Bit stuffing typically involves the addition or removal of bits to match a particular pattern.

According to at least one embodiment, system 800 may also be configured to receive as input ATM cells via, for example, one or more Utopia input ports. In one implementation, the protocol conversion and parsing logic 804 is configured to parse incoming ATM data cells (in a manner similar to that of non-ATM data) using a Utopia multiplexer. Certain information from the parser, namely a port number, ATM data and data position number (e.g., start-of-cell bit, ATM device number) is passed to a FIFO or other memory storage 808. The cell data stored in memory 808 may then be processed for channel grooming.

In specific embodiments, the frame/cell conversion logic 802 may also include a cell processor (not shown) configured to process various data parcels, including, for example, ATM cells and/or frame relay frames. The cell processor may also perform cell delineation and other functions similar to channel grooming functions performed for TDM frames. As commonly known in the field of ATM data transfer, a standard ATM cell contains 424 bits, of which 32 bits are used for the ATM cell header, eight bits are used for error correction, and 384 bits are used for the payload.

Once the incoming data has been processed and, if necessary, converted to ATM cells, the cells are input to switching logic 810. In a specific embodiment, switching logic 810 corresponds to a cell switch which is configured to route the input ATM data to an appropriate destination based on the ATM cell header (which may include a unique identifier, a port number and a device number or channel number, if input originally as serial data).

According to a specific embodiment, the switching logic 810 operates in conjunction with a scheduler 806. Scheduler 806 uses information from processor 816 which provides specific scheduling instructions and other information to be used by the scheduler for generating one or more output data streams. The processor 816 may perform these scheduling functions for each data stream independently. For example, the processor 816 may include a series of internal registers which are used as an information repository for specific scheduling instructions such as, expected addressing, channel index, QoS, routing, protocol identification, buffer management, interworking, network management statistics, etc.

Scheduler 806 may also be configured to synchronize output data from switching logic 810 to the various output ports, for example, to prevent overbooking of output ports. Additionally, the processor 816 may also manage memory 808 access requests from various system components such as those shown, for example, in FIGURES 7 and 8 of the drawings. In a specific embodiment, a memory arbiter (not shown) operating in conjunction with memory 808 controls routing of memory data to and from requesting clients using information stored in processor 816. In a specific embodiment, memory 808 includes DRAM, and memory arbiter is configured to handle the timing and execution of data access operations requested by various system components such as those shown, for example, in FIGURES 7 and 8 of the drawings..

Once cells are processed by switching logic 810, they are processed in a reverse manner, if necessary, by frame/cell conversion logic 802 and protocol conversion logic 804 before being released by system 800 via serial or TDM output ports 818 and/or parallel or Utopia output ports 820. According to a specific implementation, ATM cells are converted back to frames if the data was initially received as frames, whereas data received in ATM cell format may bypass the reverse processing of frame/cell conversion logic 802.

For purposes of illustration, the techniques of the present invention have been described with reference to their applications in ATM networks. However, it will be appreciated that the scheduling technique of the present invention may be adapted to be used in a variety of different data networks utilizing different protocols such as, for example, packet-switched networks, frame relay networks, ATM networks, etc. For example, in frame relay environments, the scheduling logic at the client entity may be configured to generate and transmit "filler" frames and/or preempt frames to the physical layer for transmission over the frame relay network. According to specific implementations, "filler" frames and/or preempt frames may be generated by inserting specific patterns of flag bytes into the output communication stream, for example, in accordance with the FRF .1.2 protocol. Such flag bytes are used to indicate that a particular portion of continuous bits (e.g. forming a frame) do not contain meaningful data, and therefore may be discarded at the physical layer of the entity receiving the communication stream.

Although several preferred embodiments of this invention have been described in detail herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to these precise embodiments, and that various changes and modifications may be effected therein by one skilled in the art without departing from the scope of spirit of the invention as defined in the appended claims.